»Лекционен курс

»ООП1 (Java)

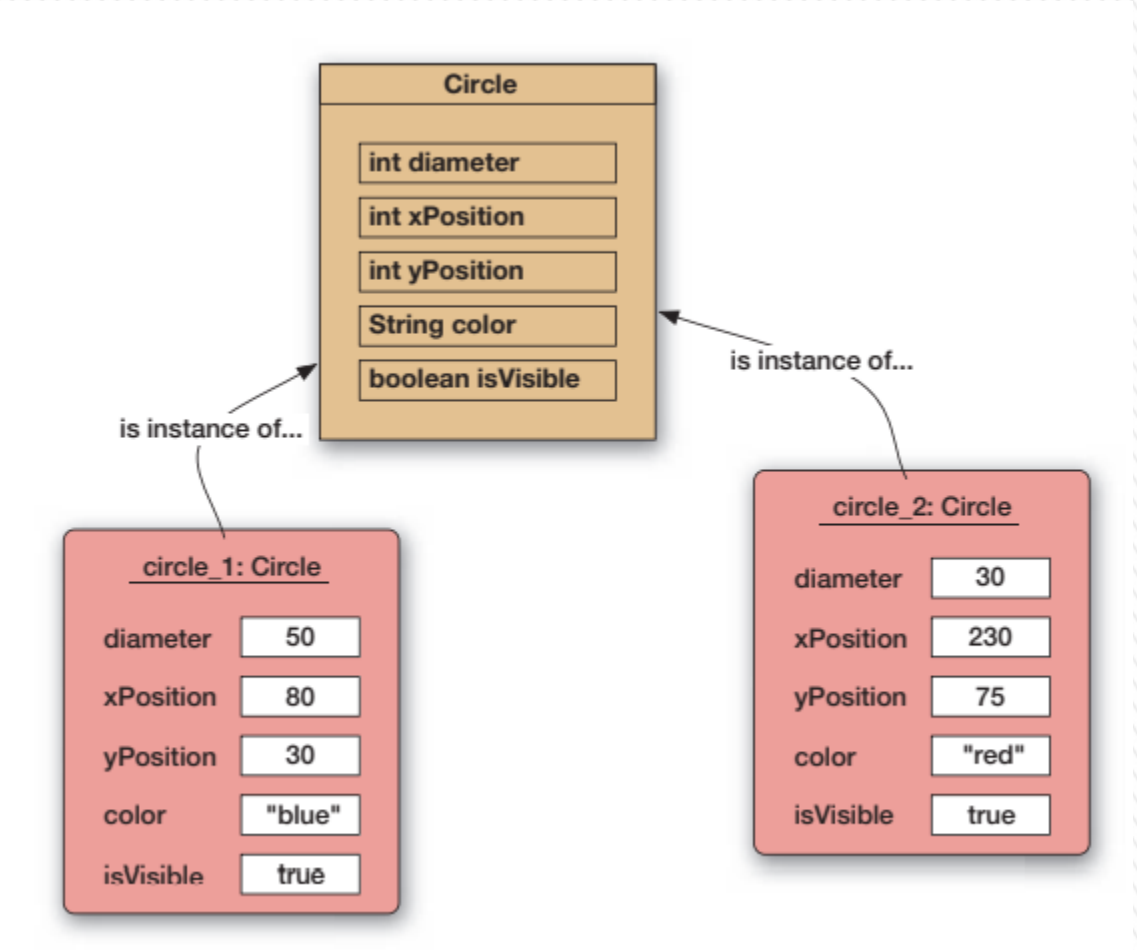
Класове, обекти, методи ›

Класове

Въвеждащ пример

Коментар?

Класове и обекти



Обектно-ориентирано програмиране

- Основен проблем в програмирането
  - Дефиниране на нови типове на обекти
- С помощта на конструкцията class можем да дефинираме нови типове от обекти, като:
  - Атрибути
  - Операции
- Два аспекта можем да разглеждаме разделено:
  - Използване на класове
  - Дефиниране на класове

Обща характеристика на клас

- Класът е тип, който определя формата на създадените чрез него обекти
- Определя както данните, така и кода, който ще работи с тези данни
- Java използва спецификация на класа за конструиране на обекти
  - Обектите са инстанции на клас
  - От конструктивна гледна точка един клас е по шаблон, определящ как да се изграждат обекти

Обща характеристика на клас

- Съществено е да се разбере, че един клас е логическа абстракция
  - Докато не е създаден обект от класа, негово физическо представяне не съществува в паметта
- Методите и променливите, представени в един клас, се наречат членове на класа
  - Членовете-данни се наричат също променливи на обект (инстанция)

Синтаксис на клас

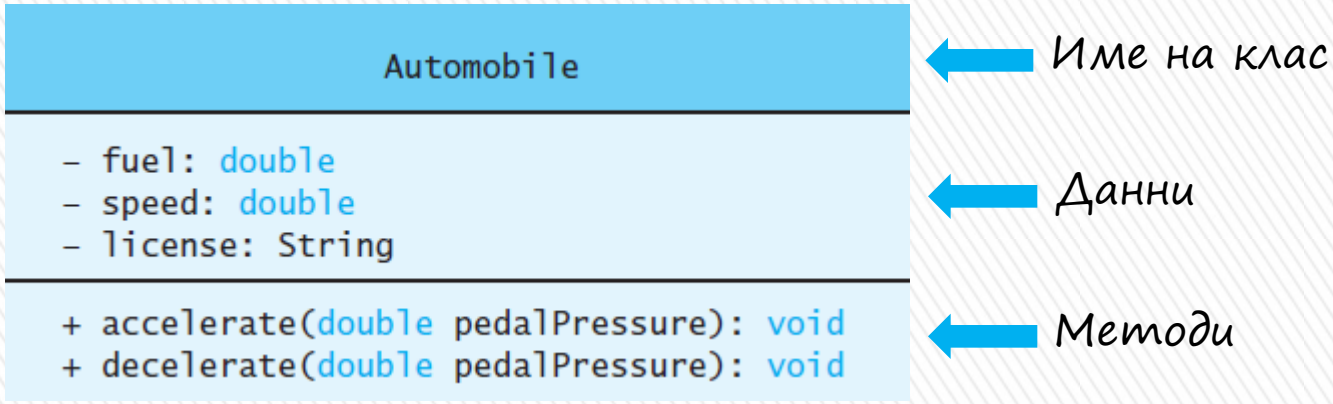
```
class name   {

        declarations

        constructor definitions

      method definitions

}
```

Символни константи и променливи

Код за създаване и инициализиране на обекти

Код за манипулиране на тези обекти

UML диаграма на клас



Обекти

Обекти

- Всеки път, когато създаваме обект от един клас, той съдържа свое собствено копие на всяка променлива на обект, декларирана в класа
  - За примера - всеки обект ще съдържа свои собствени копия на променливите fuel, speed, license
- За да получим достъп до тези променливи използваме оператора “.”
  - Този оператор свързва името на обект с името на негов член
  - Обща форма на оператора: *object.member*

Обекти

- Основен принцип: всеки обект има свое собствено копия на инстантните променливи, определени от нейния клас
- Поради това съдържанието на променливите в един обект може да се различава от съдържанието на променливите в друг обект
- Няма връзка между двата обекта, освен това, че са от същия тип

Добра практика

- Въпреки че няма налагащо синтактично правило, добра практика е един клас да дефинира само една логическа единица
  - Напр., клас, в който се съхраняват имена и телефонни номера, обикновено не съхранява и информация за фондови пазари, средни валежи, слънчеви цикли или друга несвързана информация
- Добре проектираният клас групира логически свързана информация
  - Поставяне на несвързана информация в същия клас бързо ще направи кода нещожен

Пример

Какво прави програмата?

Дефиниция на нов тип – в случая Dog

```
public class Dog {
    public String name;
    public String breed;
    public int age;
    public void writeOutput() {
        System.out.println("Име: " + name);
        System.out.println("Порода: " + breed);
        System.out.println( "Възраст (календарни години): " +
                age);
        System.out.println("Възраст (човешки години): " +
                getAgeInHumanYears());
        System.out.println();
    }
    public int getAgeInHumanYears()  {
        int humanAge = 0;
        if (age <= 2) {
            humanAge = age * 11;
        }
        else  {
            humanAge = 22 + ((age-2) * 5);
        }
        return humanAge;
    }
}
```

Пример

Какъв резултат?

Не е изпълнима – по-скоро спецификация (дефиниция) на нов тип.

```
public class Dog {
    public String name;
    public String breed;
    public int age;
    public void writeOutput() {
        System.out.println("Име: " + name);
        System.out.println("Порода: " + breed);
        System.out.println( "Възраст (календарни години): " +
                age);
        System.out.println("Възраст (човешки години): " +
                getAgeInHumanYears());
        System.out.println();
    }
    public int getAgeInHumanYears()  {
        int humanAge = 0;
        if (age <= 2) {
            humanAge = age * 11;
        }
        else  {
            humanAge = 22 + ((age-2) * 5);
        }
        return humanAge;
    }
}
```

Пример

Какво прави програмата?

Решаваме проблем – изчислява възрастта на кучета в човешки години.

```
public class DogDemo {
    public static void main(String[]
        Dog rex = new Dog();
        rex.name = "Рекс";
        rex.age = 8;
        rex.breed = "Немска овчарка";
        rex.writeOutput();
        Dog sharo = new Dog();
        sharo.name = "Шаро";
        sharo.age = 42;
        sharo.breed = "Български барак";
        System.out.println( sharo.name + " is a " +
                sharo.breed + ".");
        System.out.print( "То е на " + sharo.age +
                " години или ");
        int humanYears = sharo.getAgeInHumanYears();
        System.out.println(humanYears + " в човешки години.");
    }
}
```

Пример

Какъв резултат?

```
public class DogDemo {
    public static void main(Strin
        Dog rex = new Dog();
        rex.name = "Рекс";
        rex.age = 8;
        rex.breed = "Немска овчар
        rex.writeOutput();
        Dog sharo = new Dog();
        sharo.name = "Шаро";
        sharo.age = 42;
        sharo.breed = "Български барак";
        System.out.println( sharo.name + " is a " +
                sharo.breed + ".");
        System.out.print( "То е на " + sharo.age +
                " години или ");
        int humanYears = sharo.getAgeInHumanYears();
        System.out.println(humanYears + " в човешки години.");
    }
}
```

```
Име: Рекс
Порода: Немска овчарка
Възраст (календарни години): 8
Възраст (човешки години): 52

Шаро is a Български барак.
То е на 42 години или 222 в човешки години.

Process finished with exit code 0
```

Методи

Методи

- Променливи и методи са съставните елементи на класовете
- Капсулирането на код се извършва под формата на методи
- Методите са подпрограми, които манипулират данните, определени от класа
- В много случаи:
  - Предоставят достъп до тези данни
  - Другите части на програмата взаимодействат с класа чрез методите му

Методи

- Създаването на методи е свързано с императивното програмиране
  - Алгоритми/подалгоритми
- Основни проблеми:
  - Декларация, извикване на методи
  - Актуални, формални параметри
  - Параметри по стойност, референция

Методи

- Един метод съдържа един или повече оператора
- Добра практика е всеки метод да изпълнява само една задача
- Всеки метод има име, което се използва за извикване на метода, следвано от списък с параметри, даден в скоби (може да бъде празен)
  - Не използвайте ключови думи на Java за имена на методи
  - main() е запазен за начало на изпълнението на програмите
- Специфицира се също типът на връщания резултат
  - Може да бъде всеки валиден тип, включително типовете класове, които създаваме
  - Ако методът не връща стойност типът е void

Синтаксис на метод

```
ret-type name (parameter-list) {
                    // body of method
 }
```

Пример

- Следващият пример е предназначен да съхранява записи на застрашени видове растения
- Този клас е малко по-сложен от клас Dog, но въпреки това нарушава няколко важни принципа на ООП
- По-късно ще подобрим този пример, докато обсъждаме тези принципи
- Име на класа SpeciesFirstTry.

Какво прави програмата?

Дефиниция на нов тип – в случая SpeciesFirstTry

```
import java.util.Scanner;
public class SpeciesFirstTry {
    public String name;
    public int population;
    public double growthRate;
    public void readInput()    {
        Scanner keyboard = new Scanner(System.in);
        System.out.println("Име на вида?");
        name = keyboard.nextLine();
        System.out.println("Популация на вида?");
        population = keyboard.nextInt();
        System.out.println("Въведи индекс на нарастване " + "(% увеличение на година):");
        growthRate = keyboard.nextDouble();
    }
    public void writeOutput() {
        System.out.println("Име = " + name);
        System.out.println("Популация = " + population);
        System.out.println("Индекс нарастване = " + growthRate + "%");
    }
    public int getPopulationIn10()  {
        int result = 0;
        double populationAmount = population;
        int count = 10;
        while ((count > 0) && (populationAmount > 0)) {
            populationAmount = populationAmount + (growthRate/100) * populationAmount;
            count--;
        }
        if (populationAmount > 0)
            result = (int)populationAmount;
        return result;
    }
}
```

```
import java.util.Scanner;
public class SpeciesFirstTry {
    public String name;
    public int population;
    public double growthRate;
    public void readInput()    {
        Scanner keyboard = new Scanne
        System.out.println("Име на вида?");
        name = keyboard.nextLine();
        System.out.println("Популация на вида?");
        population = keyboard.nextInt();
        System.out.println("Въведи индекс на нарастване " + "(% увеличение на година):");
        growthRate = keyboard.nextDouble();
    }
    public void writeOutput() {
        System.out.println("Име = " + name);
        System.out.println("Популация = " + population);
        System.out.println("Индекс нарастване = " + growthRate + "%");
    }
    public int getPopulationIn10()  {
        int result = 0;
        double populationAmount = population;
        int count = 10;
        while ((count > 0) && (populationAmount > 0)) {
            populationAmount = populationAmount + (growthRate/100) * populationAmount;
            count--;
        }
        if (populationAmount > 0)
            result = (int)populationAmount;
        return result;
    }
}
```

Какъв резултат?

Не е изпълнима – по-скоро спецификация (дефиниция) на нов тип.

Дефиниция на метод

```
public void writeOutput() {
        System.out.println("Име = " + name);
        System.out.println("Популация = " + population);
        System.out.println("Индекс нарастване = " + growthRate + "%");
}
```

- Всички дефиниции на методи се появяват в дефиницията на класа, към който принадлежат – в примера в дефиницията на класа SpeciesFirstTry.
- Това означава, че методът може да се използва само с обекти от класа SpeciesFirstTry.

Дефиниция на метод

```
public void writeOutput() {
        System.out.println("Име = " + name);
        System.out.println("Популация = " + population);
        System.out.println("Индекс нарастване = " + growthRate + "%");
}
```

- Засега нашите дефиниции на методи започват с ключовата дума public, т.е. няма специални ограничения за използването на метода.
- По-нататък ще видим, че public може да бъде заменена с други модификатори, които ограничават използването на метода.
- Методът не връща стойност, т.е. тип на резултата void.

Методи, връщащи стойности

```
return Expression;
```

- Тялото на дефиниция на метод, което връща стойност, е точно като тялото на дефиниция на void метод, с изключение на това, че трябва да съдържа поне един оператор return.
- Операторът return казва, че стойността, върната от метода, е стойността на Expression.
- Изразът може да бъде всеки израз, който генерира стойност от типа, посочен в заглавието на дефиницията на метода.

Методи, връщащи стойности

```
return Expression;
```

- Когато операторът return се изпълнява, той не само връща резултат, но също така завършва изпълнението на метода.
- Ако следват повече оператори, те не се изпълняват.
- Метод, който връща стойност, може да извърши и друго действие, например четене на стойност от клавиатурата, но определено трябва да върне стойност.

В примера

```
public int getPopulationIn10()  {
        int result = 0;
        double populationAmount = population;
        int count = 10;
        while ((count > 0) && (populationAmount > 0)) {
            populationAmount = populationAmount + (growthRate/100) *

         populationAmount;
            count--;
        }
        if (populationAmount > 0)
            result = (int)populationAmount;
        return result;
    }
```

Пример

Какво прави програмата?

Решаваме проблем – изчислява бъдеща популация

```
public class SpeciesFirstTryDemo {
    public static void main(String[] ar
        SpeciesFirstTry speciesOfTheMonth = new SpeciesFirstTry();
        System.out.println("Въведи данни за вида за месец:");
        speciesOfTheMonth.readInput();
        speciesOfTheMonth.writeOutput();
        int futurePopulation =
                speciesOfTheMonth.getPopulationIn10();
        System.out.println("След 10 години популацията ще бъде "
                + futurePopulation);
        //Change the species to show how to change
        //the values of instance variables:
        speciesOfTheMonth.name = "Планински божур";
        speciesOfTheMonth.population = 10;
        speciesOfTheMonth.growthRate = 15;
        System.out.println("Новият вид за месец:");
        speciesOfTheMonth.writeOutput();
        System.out.println("След 10 години популацията ще бъде "
                + speciesOfTheMonth.getPopulationIn10());
    }
}
```

Пример

Какво за методите?

Тук се извикват декларирани в SpeciessFirstTry методи

```
public class SpeciesFirstTryDemo {
    public static void main(String[] args) {
        SpeciesFirstTry speciesOfTheMonth = new SpeciesFirstTry();
        System.out.println("Въведи данни за вида за месец:");
        speciesOfTheMonth.readInput();
        speciesOfTheMonth.writeOutput();
        int futurePopulation =
                speciesOfTheMonth.getPopulationIn10();
        System.out.println("След 10 години популацията ще бъде "
                + futurePopulation);
        //Change the species to show how to change
        //the values of instance variables:
        speciesOfTheMonth.name = "Планински божур";
        speciesOfTheMonth.population = 10;
        speciesOfTheMonth.growthRate = 15;
        System.out.println("Новият вид за месец:");
        speciesOfTheMonth.writeOutput();
        System.out.println("След 10 години популацията ще бъде "
                + speciesOfTheMonth.getPopulationIn10());
    }
}
```

Пример

Какъв резултат?

```
public class SpeciesFirstTr
    public static void main
        SpeciesFirstTry spe
        System.out.println(
        speciesOfTheMonth.r
        speciesOfTheMonth.w
        int futurePopulatio
                speciesOfTh
        System.out.println(
                + futurePop
        //Change the specie
        //the values of ins
        speciesOfTheMonth.n
        speciesOfTheMonth.p
        speciesOfTheMonth.growthRate = 15;
        System.out.println("Новият вид за месец:");
        speciesOfTheMonth.writeOutput();
        System.out.println("След 10 години популацията ще бъде "
                + speciesOfTheMonth.getPopulationIn10());
    }
}
```

```
Въведи данни за вида за месец:
Име на вида?
родопско лале
Популация на вида?
1200
Въведи индекс на нарастване (% увеличение на година):
-12,3
Име = родопско лале
Популация = 1200
Индекс нарастване = -12.3%
След 10 години популацията ще бъде 322
Новият вид за месец:
Име = Планински божур
Популация = 10
Индекс нарастване = 15.0%
След 10 години популацията ще бъде 40

Process finished with exit code 0
```

Локални променливи

```
public int getPopulationIn10()  {
        int result = 0;
        double populationAmount = population;
        int count = 10;
        while ((count > 0) && (populationAmount > 0)) {
            populationAmount = populationAmount + (growthRate/100) *

         populationAmount;
            count--;
        }
        if (populationAmount > 0)
            result = (int)populationAmount;
        return result;
    }
```

- } Методът getPopulationIn10 включва декларации на променливите populationAmount и count
- Променлива, декларирана в даден метод, се нарича локална променлива
- Нейното значение е локално, т.е. ограничено в дефиницията на метода

Локални променливи

- Да предположим, че имате два метода, дефинирани в един и същи клас
- Ако всеки метод декларира променлива със същото име бихме имали две различни променливи с едно и също име
- Всяка промяна в стойността на резултата в рамките на единия метод няма да повлияе на променливата в другия метод
  - Това би било сякаш двата метода бяха изпълнени на различни компютри или сякаш компютърът промени името на резултата в един от двата метода
- Всички променливи, декларирани в основния метод на програмата, са локални за main
  - Ако някой случайно има същото име като променлива, декларирана в друг метод, това са две различни променливи, които просто имат едно и също име

Пример

Какво прави програмата?

Дефиниция на нов тип – в случая BankAccount

```
public class BankAccount {
    public double amount;
    public double rate;
    public void showNewBalance() {
        double newAmount = amount + (rate / 100.0) * amount;
        System.out.println("С добавена лихва новата сума е " + newAmount);
    }
}
```

Пример

Какъв резултат?

Не е изпълнима – по-скоро спецификация (дефиниция) на нов тип.

```
public class BankAccount {
    public double amount;
    public double rate;
    public void showNewBalance() {
        double newAmount = amount + (rate / 100.0) * amount;
        System.out.println("С добавена лихва новата сума е " + newAmount);
    }
}
```

Пример

Какво прави програмата?

Решаваме проблем – изчислява лихва

```
public class LocalVariablesDemoProgram {
    public static void main(String[] args) {
        BankAccount myAccount = new BankAccount();
        myAccount.amount = 100.00;
        myAccount.rate = 5;
        double newAmount = 800.00;
        myAccount.showNewBalance();
        System.out.println("Иска ми се новата ми сума да беше" +
                newAmount);
    }
}
```

Пример

Какъв резултат?

```
С добавена лихва новата сума е 105.0
Иска ми се новата ми сума да беше800.0

Process finished with exit code 0
```

```
public class LocalVariablesDemoProgram {
    public static void main(String[] args) {
        BankAccount myAccount = new BankAccount();
        myAccount.amount = 100.00;
        myAccount.rate = 5;
        double newAmount = 800.00;
        myAccount.showNewBalance();
        System.out.println("Иска ми се новата ми сума да беше" +
                newAmount);
    }
}
```

Пример

```
public class BankAccount {
    public double amount;
    public double rate;
    public void showNewBalance() {
        double newAmount = amount + (rate / 100.0) * amount;
        System.out.println("С добавена лихва новата сума е " + newAmount);
    }
}
```

Не променя стойността на променливата newAmount в main

Две различни локални променливи с името newAmount

```
public class LocalVariablesDemoProgram {
    public static void main(String[] args) {
        BankAccount myAccount = new BankAccount();
        myAccount.amount = 100.00;
        myAccount.rate = 5;
        double newAmount = 800.00;
        myAccount.showNewBalance();
        System.out.println("Иска ми се новата ми сума да беше" +
                newAmount);
    }
}
```

Формални параметри

- Помислете за метода getPopulationIn10 за класа SpeciesFirstTry
- Той връща прогнозна популация на даден вид десет години в бъдещето
- Ако искаме прогнозата за пет години или петдесет години?
- Методът би бил много по-полезен, ако прогнозира популацията за даден брой години
- За да направим това, имаме нужда от някакъв начин да можем да извикваме метода с различна стойност за броя на годините
- За целта можем да използваме формални параметри

```
import java.util.Scanner;
public class SpeciesSecondTry {
    public String name;
    public int population;
    public double growthRate;
    public void readInput()   {
        Scanner keyboard = new Scanner(System.in);
        System.out.println("Име на вида?");
        name = keyboard.nextLine();
        System.out.println("Популация на вида?");
        population = keyboard.nextInt();
        System.out.println("Въведи индекс на нарастване " + "(% увеличение на година):");
        growthRate = keyboard.nextDouble();
    }
    public void writeOutput() {
        System.out.println("Име = " + name);
        System.out.println("Популация = " + population);
        System.out.println("Индекс нарастване = " + growthRate + "%");
    }
    public int predictPopulation(int years) {
        int result = 0;
        double populationAmount = population;
        int count = years;
        while ((count > 0) && (populationAmount > 0)) {
            populationAmount = (populationAmount +
                    (growthRate / 100) * populationAmount);
            count--;
        }
        if (populationAmount > 0)
            result = (int)populationAmount;
        return result;
    }
}
```

Формален параметър (параметър) years

Актуални параметри

```
import java.util.Scanner;
public class SpeciesSecondTryDemo {
    public static void main(String[] args) {
        Scanner keyboard = new Scanner(System.in);
        SpeciesSecondTry mySpecies = new SpeciesSecondTry();
        System.out.println("Въведи данни за вида:");
        mySpecies.readInput();
        mySpecies.writeOutput();
        System.out.println("Въведи годините за прогноза:");
        int projectedYears = keyboard.nextInt();
        int futurePopulation = mySpecies.
                predictPopulation(projectedYears);
        System.out.println("След " + projectedYears + " години, ");
        System.out.println("популацията ще бъде "+ futurePopulation);
    }
}
```

- *Когато извикваме този метод му придаваме стойност (актуален параметър, аргумент), с която искаме да заменим формалния параметър в изпълнението на метода*
- *Формалните и актуалните параметри трябва да са от един тип, поставяни в еднакъв ред*

Актуални параметри

Какъв резултат?

```
Въведи данни за вида:
Име на вида?
бор
Популация на вида?
123
Въведи индекс на нарастване (% увеличение на година):
-12,4
Име = бор
Популация = 123
Индекс нарастване = -12.4%
Въведи годините за прогноза:
5
След 5 години,
популацията ще бъде 63

Process finished with exit code 0
```

```
import java.util.Scanner;
public class SpeciesSecondTryDemo {
    public static void main(String[
        Scanner keyboard = new Scan
        SpeciesSecondTry mySpecies
        System.out.println("Въведи
        mySpecies.readInput();
        mySpecies.writeOutput();
        System.out.println("Въведи годините за прогноза:");
        int projectedYears = keyboard.nextInt();
        int futurePopulation = mySpecies.
                predictPopulation(projectedYears);
        System.out.println("След " + projectedYears + " години, ");
        System.out.println("популацията ще бъде "+ futurePopulation);
    }
}
```

Предаване на параметри

- Принципно съществуват две възможности за предаване на параметри на методите:
  - По стойност (call-by-value механизъм)
  - По референция (call-by-reference механизъм)

Предаване параметри по стойност

- При предаване на параметри използва само стойността на аргумента
- В Java това е единственият механизъм за предаване на параметри от примитивен тип, като int, double и char
- Формалните параметри, които се появява в дефиницията на метода, се разглеждата като локални променливи
- Те се инициализират със стойността на аргумента (актуалния параметър), даден в извикването на метода

- Параметрите-обекти използват предаване по референция

Предаване параметри по референция

- Параметрите-обекти се предават по референция
- В Java се различава дали е предаден примитивен тип или референтен тип
- Когато създаваме променлива от тип клас, създаваме само реферанция към обект
- По този начин, когато предадем актуален параметър на метод референцията ще се отнася до същия обект като този, посочен от аргумента
- Това на практика означава, че обектите действат така, сякаш са предадени на методи чрез референция
- Промените в обекта в метода засягат обекта, използван като аргумент.

Обобщение

- Възможно е да се предадат една или повече стойности на метод, когато той се извиква
- Стойността, предавана на метод, се нарича аргумент
- Променливата вътре в метода, която получава аргумента, се нарича параметър
  - Параметрите се декларират в скобите, които следват името на метода
- Синтаксисът на декларацията на параметрите е същият като този, използван за променливите
  - Параметърът е в обхвата на неговия метод и освен специалната му задача да получава аргумент, той действа като всяка друга локална променлива

Класове и стандартни типове данни

- Това, което правим в ООП е създаване на нови типове данни
- Едно число е също тип – има определени характеристики и поведение
  - Разлика: в ООП дефинираме класове, подходящи за решаване на дадена задача, вместо да ни бъде наложено да използваме съществуващ тип данни, който е бил проектиран да представя единица в машината
- Добавяйки нови типове данни, специфични за решаване на различни задачи, ние разширяваме езика за програмиране

Обобщение

Обектите имат:

- Състояние
- Поведение

Състояние: конфигурация на данните на обекта в определен момент.

Поведение: интерфейсът на обекта, т.е. публичните методи на обекта.

Публичните методи на класа определят неговия интерфейс!

Обобщение

Контролират ли обектите състоянията си или могат ли да бъдат променяно състоянието на един обект при работа с него?

Да

Контролират ли обектите поведението си или могат ли да бъде променяно поведението на един обект при работа с него?

Не


Благодаря за вниманието!